Docket No.   RSW9-2000-0089-US1

METHOD AND APPARATUS FOR SECURING SESSION INFORMATION OF USERS IN A WEB APPLICATION SERVER ENVIRONMENT

BACKGROUND OF THE INVENTION

1. Technical Field:

5   The present invention relates to an improved data processing system and, in particular, to a method and apparatus for managing sessions in a client/server environment. Still more particularly, the present invention provides a method and apparatus for securing

10 hypertext transfer protocol sessions.

2. Description of Related Art:

The worldwide network of computers commonly known as the "Internet" has seen explosive growth in the last several years. Mainly, this growth has been fueled by the

15 introduction and widespread use of so-called "web browsers," which enable simple graphical user interface-based access to network servers, which support documents formatted as so-called "web pages." A browser is a program that is executed on a graphical user interface (GUI) in a

20 client computer. The browser allows a user to seamlessly load documents from a server via the Internet and display them by means of the GUI. These documents are commonly formatted using markup language protocols, such as hypertext markup language (HTML).

Docket No.   RSW9-2000-0089-US1

　　　　The client and the web server typically communicate using hypertext transport protocol (HTTP). However, when a client is accessing sensitive information from a web server, a secure protocol may be used. Hypertext transport

5 protocol secure is the protocol for accessing a secure Web server. Using HTTPS in the uniform resource locator (URL) instead of HTTP directs the message to a secure port number rather than the default Web port number of 80. The session is then managed by a security protocol. Secure sockets

10 layer is the leading security protocol on the Internet. When a session is started in SSL, the browser sends its public key to the server so that the server can securely send a secret key to the browser. The browser and server exchange data via secret key encryption during that

15 session.

　　　　However, HTTP is a stateless protocol. Therefore, every request from an HTTP client to an HTTP server is a new request and no state is maintained between requests. Conventionally, HTTP cookies are used to maintain a client-

20 side state whereas sessions are used to manage the state information on the server side. A cookie is data created by a web server that is stored on a client computer. A cookie is used to keep track of a user's patterns and preferences and, with the cooperation of the Web browser,

25 is stored within the client computer. Cookies contain a range of URLs for which they are valid. When the browser

Docket No. RSW9-2000-0089-US1

encounters those URLs again, it sends the appropriate

cookies to the Web server.

A session is used to track the activities of a user.

For example, a session may be created to allow a user to

5 add items to a "shopping cart" using a plurality of

individual requests. A session may also allow a user to

use a web interface to search a database. Web interfaces

may also be used to control equipment from remote

locations. As web interfaces become increasingly popular,

10 the security of sessions used to manage multiple

transactions by individual clients becomes exceedingly

important. Normally, a session is created on the server

side. To associate a session with a user, a random number,

referred to as a session identification (ID), is generated

15 and associated with the user. The session ID is sent back

to the browser as a cookie or through a URL rewriting

mechanism.

When an HTTP request is received, the server verifies

if a session ID is present. If an ID is present, the

20 related session data is retrieved and the request is

processed based on the session data. However, the server

cannot verify that the user submitting the request is the

same user to whom the session ID was originally assigned.

Hence, a security loophole exists where an unauthorized

25 user may submit a valid session ID. The session ID may be

obtained by repeatedly submitting requests with potential

Docket No.   RSW9-2000-0089-US1

session identifications until access is granted.
Alternatively, the ID may be "sniffed" from the network by
monitoring data traffic flow.   The session ID may be
obtained in this manner when a request is transmitted

5   through an unsecure protocol, such as HTTP, as opposed to a
secure protocol, such as HTTPS or SSL.   In many web
application server products, the security of session
information is tied only to the randomness of the session
ID under the assumption that the bit length of the number

10   is high enough to prevent an unauthorized user from
generating the same number in a short period of time.
However, the likelihood of hijacking the session ID is not
ruled out completely.

Therefore, it would be advantageous to have an

15   improved method and apparatus for securing session
information of users in a web application server
environment.

Docket No.   RSW9-2000-0089-US1

SUMMARY OF THE INVENTION

The present invention associates a session with a user's authentication credentials. A session is created using credentials, such as the user's identification and password, a security token, or a digital certificate. Subsequent requests must be submitted with the session ID and the user credentials associated with the session. Thus, the session information is as secure as the server infrastructure.

Docket No.   RSW9-2000-0089-US1

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of

5 use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 is a pictorial representation of a distributed

10 data processing system in which the present invention may be implemented;

Figure 2 is a block diagram of a data processing system that may be implemented as a server in accordance with a preferred embodiment of the present invention;

15 **Figure 3** is a block diagram illustrating a data processing system in which the present invention may be implemented;

Figure 4 depicts a specific client/server arrangement according to a preferred embodiment of the present

20 invention;

Figure 5A is a flowchart illustrating the creation of a session in accordance with a preferred embodiment of the present invention;

Figure 5B is a flowchart illustrating the creation of

25 a session for a user with an established account on the

Docket No. RSW9-2000-0089-US1

server in accordance with a preferred embodiment of the

present invention; and

Figure 5C is a flowchart illustrating the management

of a request in an established session according to a

5 preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

With reference now to the figures, Figure 1 depicts a

pictorial representation of a distributed data processing

system in which the present invention may be implemented.

10 Distributed data processing system 100 is a network of

computers in which the present invention may be implemented.

Distributed data processing system 100 contains a network

102, which is the medium used to provide communications

links between various devices and computers connected

15 together within distributed data processing system 100.

Network 102 may include permanent connections, such as wire

or fiber optic cables, or temporary connections made through

telephone connections.

In the depicted example, a server 104 is connected to

20 network 102 along with storage unit 106. In addition,

clients 108, 110, and 112 also are connected to network 102.

These clients 108, 110, and 112 may be, for example,

personal computers or network computers. For purposes of

this application, a network computer is any computer,

Docket No.   RSW9-2000-0089-US1

coupled to a network, which receives a program or other
application from another computer coupled to the network.
In the depicted example, server **104** provides data, such as
boot files, operating system images, and applications to

5 clients **108-112**.   Clients **108**, **110**, and **112** are clients to
server **104**.   Distributed data processing system **100** may
include additional servers, clients, and other devices not
shown.   In the depicted example, distributed data processing
system **100** is the Internet with network **102** representing a

10 worldwide collection of networks and gateways that use the
TCP/IP suite of protocols to communicate with one another.
At the heart of the Internet is a backbone of high-speed
data communication lines between major nodes or host
computers, consisting of thousands of commercial,

15 government, educational and other computer systems that
route data and messages.   Of course, distributed data
processing system **100** also may be implemented as a number of
different types of networks, such as for example, an
intranet, a local area network (LAN), or a wide area network

20 (WAN). **Figure 1** is intended as an example, and not as an
architectural limitation for the present invention.

Referring to **Figure 2**, a block diagram of a data
processing system that may be implemented as a server, such
as server **104** in **Figure 1**, is depicted in accordance with a

25 preferred embodiment of the present invention.   Data
processing system **200** may be a symmetric multiprocessor

Docket No.   RSW9-2000-0089-US1

(SMP) system including a plurality of processors **202** and **204**
connected to system bus **206**. Alternatively, a single
processor system may be employed. Also connected to system
bus **206** is memory controller/cache **208**, which provides an
5 interface to local memory **209**. I/O bus bridge **210** is
connected to system bus **206** and provides an interface to I/O
bus **212**. Memory controller/cache **208** and I/O bus bridge **210**
may be integrated as depicted.

Peripheral component interconnect (PCI) bus bridge **214**
10 connected to I/O bus **212** provides an interface to PCI local
bus **216**. A number of modems may be connected to PCI bus
216. Typical PCI bus implementations will support four PCI
expansion slots or add-in connectors. Communications links
to network computers **108-112** in **Figure 1** may be provided
15 through modem **218** and network adapter **220** connected to PCI
local bus **216** through add-in boards.

Additional PCI bus bridges **222** and **224** provide
interfaces for additional PCI buses **226** and **228**, from which
additional modems or network adapters may be supported. In
20 this manner, data processing system **200** allows connections
to multiple network computers. A memory-mapped graphics
adapter **230** and hard disk **232** may also be connected to I/O
bus **212** as depicted, either directly or indirectly.

Those of ordinary skill in the art will appreciate that
25 the hardware depicted in **Figure 2** may vary. For example,
other peripheral devices, such as optical disk drives and

Docket No.   RSW9-2000-0089-US1

the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

5   The data processing system depicted in **Figure 2** may be, for example, an IBM RISC/System 6000 system, a product of International Business Machines Corporation in Armonk, New York, running the Advanced Interactive Executive (AIX) operating system.

10   With reference now to **Figure 3**, a block diagram illustrating a data processing system in which the present invention may be implemented. Data processing system **300** is an example of a client computer. Data processing system **300** employs a peripheral component interconnect (PCI) local bus

15 architecture. Although the depicted example employs a PCI bus, other bus architectures such as Accelerated Graphics Port (AGP) and Industry Standard Architecture (ISA) may be used. Processor **302** and main memory **304** are connected to PCI local bus **306** through PCI bridge **308**. PCI bridge **308**

20 also may include an integrated memory controller and cache memory for processor **302**. Additional connections to PCI local bus **306** may be made through direct component interconnection or through add-in boards. In the depicted example, local area network (LAN) adapter **310**, SCSI host bus

25 adapter **312**, and expansion bus interface **314** are connected to PCI local bus **306** by direct component connection. In

Docket No.   RSW9-2000-0089-US1

contrast, audio adapter **316**, graphics adapter **318**, and

audio/video adapter **319** are connected to PCI local bus **306**

by add-in boards inserted into expansion slots.   Expansion

bus interface **314** provides a connection for a keyboard and

5 mouse adapter **320**, modem **322**, and additional memory **324**.

Small computer system interface (SCSI) host bus adapter **312**

provides a connection for hard disk drive **326**, tape drive

328, and CD-ROM drive **330**.   Typical PCI local bus

implementations will support three or four PCI expansion

10 slots or add-in connectors.

An operating system runs on processor **302** and is used

to coordinate and provide control of various components

within data processing system **300** in **Figure 3**.   The

operating system may be a commercially available operating

15 system, such as Windows 2000, which is available from

Microsoft Corporation.   An object oriented programming

system such as Java may run in conjunction with the

operating system and provides calls to the operating system

from Java programs or applications executing on data

20 processing system **300**.   "Java" is a trademark of Sun

Microsystems, Inc.   Instructions for the operating system,

the object-oriented operating system, and applications or

programs are located on storage devices, such as hard disk

drive **326**, and may be loaded into main memory **304** for

25 execution by processor **302**.

Docket No.   RSW9-2000-0089-US1

Those of ordinary skill in the art will appreciate that the hardware in **Figure** 3 may vary depending on the implementation. Other internal hardware or peripheral devices, such as flash ROM (or equivalent nonvolatile

5 memory) or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in **Figure** 3. Also, the processes of the present invention may be applied to a multiprocessor data processing system.

For example, data processing system **300**, if optionally

10 configured as a network computer, may not include SCSI host bus adapter **312**, hard disk drive **326**, tape drive **328**, and CD-ROM **330**, as noted by dotted line **332** in **Figure** 3 denoting optional inclusion. In that case, the computer, to be properly called a client computer, must include some

15 type of network communication interface, such as LAN adapter **310**, modem **322**, or the like. As another example, data processing system **300** may be a stand-alone system configured to be bootable without relying on some type of network communication interface, whether or not data

20 processing system **300** comprises some type of network communication interface. As a further example, data processing system **300** may be a Personal Digital Assistant (PDA) device, which is configured with ROM and/or flash ROM in order to provide non-volatile memory for storing

25 operating system files and/or user-generated data.

Docket No.   RSW9-2000-0089-US1

The depicted example in **Figure 3** and above-described examples are not meant to imply architectural limitations. For example, data processing system **300** also may be a notebook computer or hand held computer in addition to

5 taking the form of a PDA. Data processing system **300** also may be a kiosk or a Web appliance.

With reference now to **Figure 4**, a specific client/server arrangement is presented according to a preferred embodiment of the present invention. A network,

10 such as Internet **402**, connects a server **404**, a first client **406**, and a second client **408**. In the depicted example, the network is the Internet; however, any network that supports HTTP protocol or uses a protocol that may use sessions may be implemented.

15   Server **404** is connected to storage **410**. Similarly, first client **406** is connected to storage **412** and second client **408** is connected to storage **414**. When the first client sends a request to the server, session **420** is created and session-ID **422** is generated. Session **420** is

20 stored in a data structure referred to as a session object. Once the session is created, the server fulfills the request and session ID **422** is returned, along with the content requested, to the first client as cookie **430**, which is stored at the client. The session ID may also be sent

25 to the client using URL rewriting or other known techniques.

Docket No.   RSW9-2000-0089-US1

According to a preferred embodiment of the present
invention, client **406** sends credentials **432** to server **404**
with the request. The server then authenticates the
credentials and creates session **420** in response to the
5 authentication being successful. The credentials are then
stored in association with the session as credentials **424**
to be used for subsequent requests in the session. The
credentials may be the user's identification and password,
a security token issued by the server, or a digital
10 certificate issued by a trusted third party. The user's
identification and password may be stored in storage **412**,
414 or entered by the user at an input device. A security
token or certificate may be stored in storage **412**, **414**,
entered by the user at an input device, or entered by use
15 of a card or bar code. A person of ordinary skill in the
art will recognize that other credentials may also be used.

In a similar manner, when second client **408** sends a
request to the server, session **425** is created and session
ID **426** is generated. Session **425** is stored in a data
20 structure in storage **410**. Session ID **426** is returned to
the second client as cookie **434**, which is stored at the
client. The second client sends credentials **436** to the
server with the request. The server then authenticates the
credentials and creates session **425** in response to the
25 authentication being successful. The credentials are then

Docket No.   RSW9-2000-0089-US1

stored in association with the session as credentials **428**
to be used for subsequent requests in the session.

Storage **410** may also have stored therein accounts **440**
associated with established users. Sessions may be used to
5 establish an account for the associated user. The server
may also use the received credentials to associate an
established account with the session. An account may
contain bank account information, a user's shopping cart
that stores products selected for purchase, an e-mail
10 account, display preferences for a portal or other web
site, on-going searches of a database, age verification
information for access to restricted material, etc.

In a subsequent request, client **406** submits the
request along with the session ID and credentials. Server
15 **404** authenticates the session ID and credentials and
continues session **420** if authentication is successful.
Encryption may also be used to protect the session ID and
credentials from being obtained by malicious users. Also,
the request including the session ID and credentials may be
20 sent over HTTPS to avoid session ID and credential
hijacking.

With reference now to **Figure 5A**, a flowchart is shown
illustrating the creation of a session in accordance with a
preferred embodiment of the present invention. The client
25 initiates a request (step **502**), which requires
authentication (step **504**). The client authenticates (step

Docket No.   RSW9-2000-0089-US1

506) by supplying credentials, which may be the user's
identification and password, a security token issued by the
server, or a certificate issued by a trusted third party.
The request and credentials are then submitted to the

5   server, which checks authentication (step 508).

A determination is made as to whether authentication
is successful (step 510). If authentication is not
successful, a notification that access is denied is
returned to the client, which presents the notification to

10   the user (step 512). If authentication is successful in
step 510, the server generates a random session ID (step
514) and fulfills the request (step 516).

The session ID is returned to the client as a cookie,
along with the content associated with the fulfilled

15   request. Thereafter the client stores the session ID
cookie (step 518) and the user accesses the content (step
520).

Turning now to Figure 5B, a flowchart is shown
illustrating the creation of a session for a user with an

20   established account on the server in accordance with a
preferred embodiment of the present invention. The client
initiates a request (step 522), which requires
authentication (step 524). The client authenticates (step
526) by supplying credentials, which may be the user's

25   identification and password, a security token issued by the
server, or a certificate issued by a trusted third party.

Docket No.   RSW9-2000-0089-US1

The request and credentials are then submitted to the
server, which checks authentication (step **528**).

A determination is made as to whether authentication
is successful (step **530**). If authentication is not
5 successful, a notification that access is denied is
returned to the client, which presents the notification to
the user (step **532**). If authentication is successful in
step **530**, the server retrieves a user account associated
with the credentials (step **533**), generates a random session
10 ID (step **534**), and fulfills the request (step **536**).

The session ID is returned to the client as a cookie,
along with the content associated with the fulfilled
request. Thereafter the client stores the session ID
cookie (step **538**) and the user accesses the content (step
15 **540**).

Turning now to **Figure 5C**, a flowchart is shown
illustrating the management of a request in an established
session according to a preferred embodiment of the present
invention. The client initiates a request (step **542**) and
20 submits the request along with the session ID and
credentials. The server checks the credentials (step **544**)
and determines whether authorization is successful (step
546). If authorization is not successful, a notification
that access is denied is returned to the client, which
25 presents the notification to the user (step **548**). If

Docket No.   RSW9-2000-0089-US1

authorization is successful in step **546**, a determination is

made as to whether the session ID is valid (step **550**).

    If the session ID is not valid, a notification that

access is denied is returned to the client, which presents

5 the notification to the user (step **552**). If the session ID

is valid in step **550**, the server retrieves the session

object (step **554**), compares the credentials from the

request with the credentials in the session object

authentication (step **556**), and determines whether the

10 credentials match (step **558**).

    If the credentials do not match, a notification that

access is denied is returned to the client, which presents

the notification to the user (step **560**). If the

credentials do match in step **558**, the server fulfills the

15 request (step **562**) and returns content associated with the

fulfilled request to the client. Thereafter, the user

accesses the content (step **564**).

    The present invention authenticates a user's

credentials before creating an HTTP session. The present

20 invention then associates the session with the credentials.

Subsequent requests must be submitted with the session ID

and the user credentials to be associated with the session.

Therefore, an unauthorized user that has obtained a session

ID by monitoring traffic in the network or by chance cannot

25 gain access to sensitive content associated with the

session without also obtaining the credentials. The

Docket No.   RSW9-2000-0089-US1

present invention solves the disadvantages of the prior art

by requiring credentials for each request within a session.

Thus, the present invention provides increased security

against session hijacking.

5   It is important to note that while the present

invention has been described in the context of a fully

functioning data processing system, those of ordinary skill

in the art will appreciate that the processes of the

present invention are capable of being distributed in the

10 form of a computer readable medium of instructions and a

variety of forms and that the present invention applies

equally regardless of the particular type of signal bearing

media actually used to carry out the distribution.

Examples of computer readable media include recordable-type

15 media such a floppy disc, a hard disk drive, a RAM, and CD-

ROMs and transmission-type media such as digital and analog

communications links.

The description of the present invention has been

presented for purposes of illustration and description, but

20 is not intended to be exhaustive or limited to the

invention in the form disclosed. Many modifications and

variations will be apparent to those of ordinary skill in

the art. For example, HTTP is used as the protocol in the

depicted examples, but the present invention may be

25 implemented using other protocols. Also, in the depicted

examples, the session_ID is sent from the server to the

Docket No.   RSW9-2000-0089-US1

client in a cookie; however, the session ID may be sent by URL rewriting or other known techniques. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and

5 to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.